

User's Manual

ACTI✓ALIGNMENT

Don't Bother About Location

Microsoft, Windows, Windows NT, Windows 2000, Windows XP, Visual Basic, Microsoft .NET, Visual C++, Visual C#, and ActiveX are either trademarks or registered trademarks of Microsoft Corporation.

All other nationally and internationally recognized trademarks and tradenames are hereby recognized.

Copyright © 2001-2008 by MVTec Software GmbH, München, Germany



Edition 1	April 2001	(ActivVisionTools 1.3)
Edition 2	September 2001	(ActivVisionTools 2.0)
Edition 3	November 2002	(ActivVisionTools 2.1)
Edition 4	January 2005	(ActivVisionTools 3.0)
Edition 5	February 2006	(ActivVisionTools 3.1)
Edition 6	May 2008	(ActivVisionTools 3.2)

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of the publisher.

More information about ActivVisionTools can be found at:

<http://www.activ-vision-tools.com>

How to Read This Manual

This manual explains how to use `ActivAlignment` to automatically align regions of interest (ROIs) of other `ActivVisionTools` to moving parts. It describes the functionality of `ActivAlignment` and its cooperation with other `ActivVisionTools` with Visual Basic examples. Before reading this manual, we recommend to read the manual [Getting Started with ActivVisionTools](#), which introduces the basic concepts of `ActivVisionTools` and the [User's Manual for ActivView](#) to learn how to load and display images.

To follow the examples actively, first install and configure `ActivVisionTools` as described in the manual [Getting Started with ActivVisionTools](#). For each example in this manual, there is a corresponding Visual Basic project; these projects can be found in the subdirectory `examples\manuals\activalignment` of the `ActivVisionTools` base directory you specified during installation (default: `C:\Program Files\MVTec\ActivVisionTools`). Of course, you can also create your own Visual Basic projects from scratch.

We recommend to **create a private copy of the example projects** because by experimenting with the projects, you also change their *state*, which is then automatically stored in the so-called description files (extension `.dsc`) by `ActivVisionTools`. Of course, you can restore the state of a project by retrieving the corresponding description file from the CD.



Contents

1	About ActivAlignment	1
1.1	Introducing ActivAlignment	2
1.2	The Sub-Tools of ActivAlignment	5
2	Using ActivAlignment	7
2.1	Selecting the Alignment Anchor	8
2.2	Teaching the Alignment Anchor	10
2.3	Finding the Anchor Again	12
3	Combining ActivAlignment with other ActivVisionTools	15
3.1	Aligning the ROIs for ActivMeasure	16
3.2	Evaluating Results	18
3.3	Output of Results	20
4	Tips & Tricks	23
4.1	Adapting the Display of Results	24
4.2	Configuring the Two Execution Modes	26
4.3	Accessing Results Via the Programming Interface	28

Chapter 1

About ActivAlignment

This chapter will introduce you to the features and the basic concepts of the *change detector* ActivAlignment. It gives an overview about ActivAlignment’s *master tool* and its *support tools*, which are described in more detail in [chapter 2](#) on page [7](#) and [chapter 3](#) on page [15](#).

1.1	Introducing ActivAlignment	2
1.2	The Sub-Tools of ActivAlignment	5

1.1 Introducing ActivAlignment

With the help of ActivAlignment, you can align the regions of interest (ROIs) of other Activ-VisionTools to moving parts. In other words, you can “anchor” an ROI to a certain object in the image; if the object moves from image to image, the ROI moves with it (see [figure 1.1](#)). To anchor an ROI to an object, you must first teach ActivAlignment how to recognize the object in an image. For this, you need one so-called *reference image*; in it, you select a part that is to act as the *alignment anchor*, i.e., by which the object can be recognized. In [figure 1.1](#), the imprint on the IC acts as the anchor.

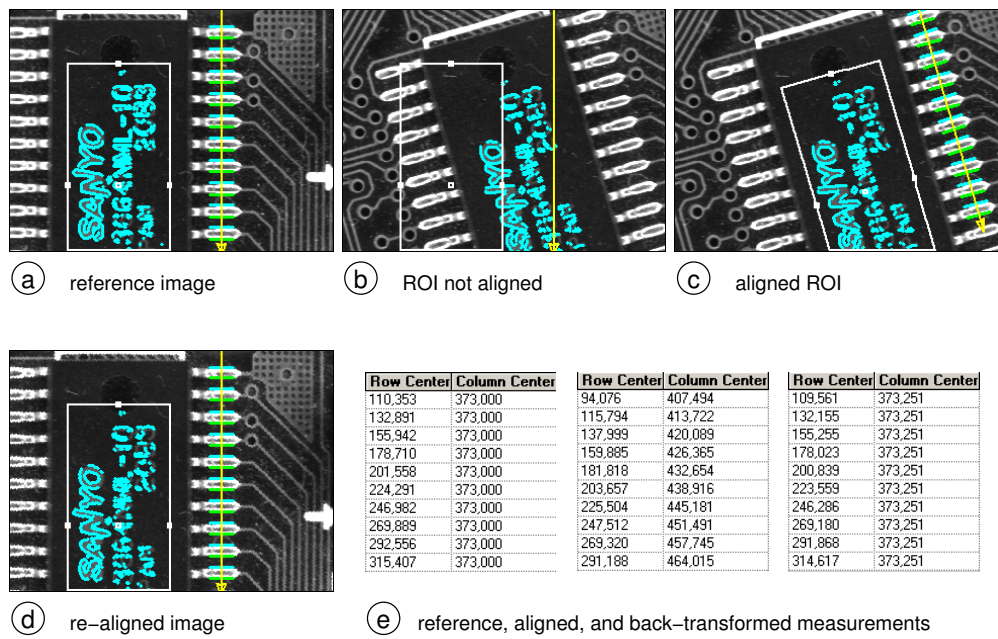


Figure 1.1: Aligning a measuring ROI to an IC.

ActivAlignment is used in two phases: In the *training phase* you specify the alignment anchor; this is typically done in the *configuration mode*. In the *application phase* (typically in the *application mode*), ActivAlignment automatically searches for the object in the current image and aligns the ROIs of other ActivVisionTools accordingly.

The Reference Image

To train ActivAlignment only one prototype image is required, where the part you want to analyze via machine vision is in its default position. In this so-called reference image you select a region in the image that is to serve as the alignment anchor (see below) and create the ROIs that are to be aligned (see [figure 1.1a](#)). The position of the anchor and the ROIs in the reference image is then stored; when ActivAlignment finds the anchor in another image, the stored information is used to align the ROIs.

ActivAlignment also offers to transform data of the current image back into the reference image. For example, you can transform the current image shown in [figure 1.1c](#) on page 2 to get the image shown in [figure 1.1d](#) on page 2; this kind of visualization helps to spot the differences between images more quickly. Alternatively, you can leave the visualization as it is and just transform the calculated results (see [figure 1.1e](#) on page 2, with the measured features being the centers of the edge pairs). To see the effect of the transformation compare the values of Column Center in [figure 1.1e](#) on page 2. This mechanism could also be called “making the features invariant” regarding position and orientation changes.

Selecting the Alignment Anchor

In order to assure a robust alignment of ROIs, you should exercise some care in selecting the part of the image that is to serve as the alignment anchor: First of all, the anchor must be *visible*, at least partially, for all possible positions and orientations of the part. Secondly, it should be *unique*, i.e., not to be confused with other parts of the image. Thirdly, the appearance of the anchor should not change from image to image, i.e., it should be *stable*. The better these criteria are met, the more you can restrict the *search space* (see below) and thereby speed up the alignment process.

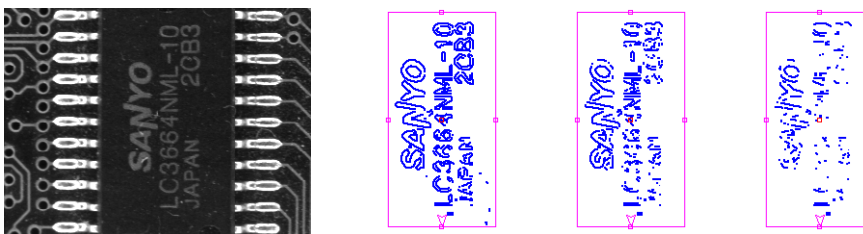


Figure 1.2: The original image and the contours on the IC with a contrast higher than 34, 41, 48 gray levels.

In order to judge whether an image part forms a suitable anchor you need to know something about the underlying matching process. In its current version, ActivAlignment uses the so-called *shape-based matching*. In contrast to the classical pattern matching techniques, which are based on comparing pixel values directly, shape-based matching evaluates the shape of an object in

form of its contours (see [figure 1.2](#)). The main advantage is that contours characterize an object in a compressed way, which significantly speeds up the matching process. Furthermore, the contours are robust against changing illumination conditions and are still recognizable in the presence of clutter (noise) or when the object is partially occluded.

Armed with this information let's take a closer look at some of the criteria for alignment anchors. The requested uniqueness has two implications: Obviously, an anchor should appear only once in the image. Applied to the example in [figure 1.2](#) on page 3 this means that you should not anchor an ROI to a single character 'L' as it appears twice in the second row. The second implication of uniqueness is anchors should not show any rotational symmetry like for example the character 'S' or 'N'. Otherwise, they could be "confused with themselves", i.e., detected at the correct position but rotated by $\pm 180^\circ$. Another solution for this problem is to restrict the range of orientation for which the anchor is trained to below $\pm 180^\circ$. Similarly, if you want to use a cross-shaped anchor you must restrict the range of orientation to below $\pm 90^\circ$.

The request for stable anchors is already assured to a high degree by the shape-based matching algorithm itself, which is robust against changing illumination, clutter, and even partial occlusion. You can further enhance stability by selecting only "significant" contours, i.e., contours with a high contrast (see [figure 1.2](#) on page 3).

The Search Space

The concept of a search space is closely related to the question how much the position of the anchor will differ from the reference image. For example, if the object to be inspected moves only slightly, ActivAlignment can restrict its search for the anchor to a small part in the image. This also holds for the possible orientation of an object. The advantage of limiting the search space like this is of course a considerable speedup. Moreover, an anchor needs to be unique only within the search space.

But not only the position and orientation, also the appearance of an object may change from image to image, for example due to noise or partial occlusion. Again, it is advantageous to tell ActivAlignment how much the appearance may change in order to restrict the search.

Results

If ActivAlignment finds the anchor in the current image, ROIs of other ActivVisionTools are automatically aligned to it. However, what happens if ActivAlignment fails to find the anchor? For this case, you can choose between the following reactions: First, you can ignore the failure; this means that the other tools will work on the ROIs as they were specified in the reference image and probably yield meaningless results. Secondly, you can choose to stop the application immediately, i.e., before executing any other tools. Finally, ActivAlignment can create a so-called error result. In this case, the application continues but the other tools do not perform any image processing on the ROIs that should have been aligned.

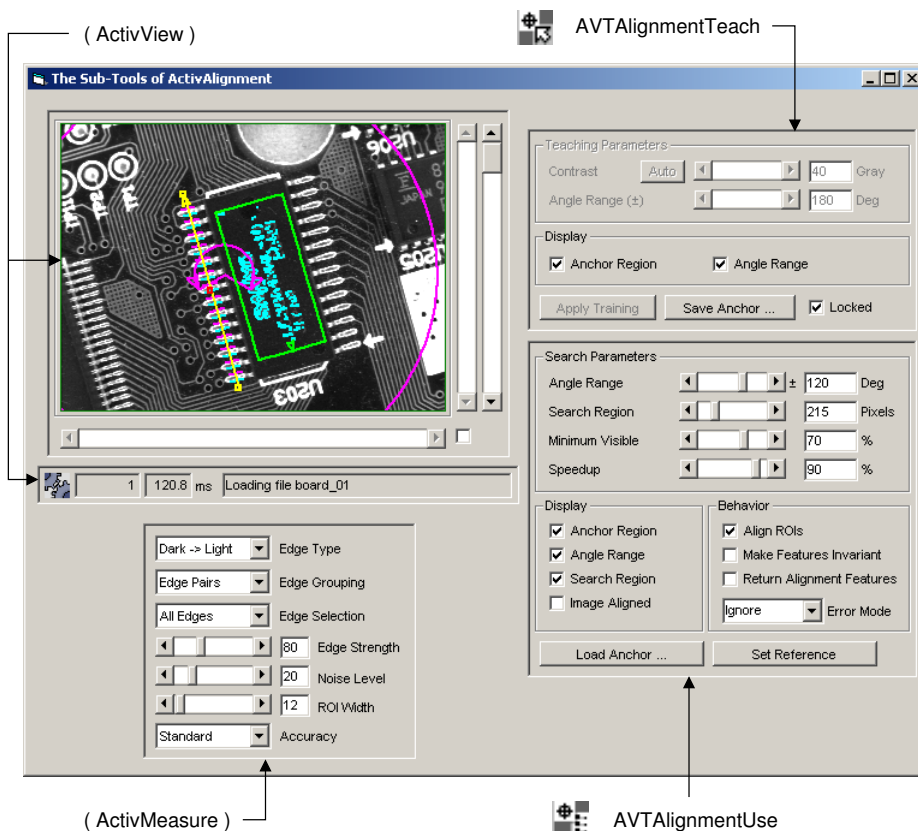


Figure 1.3: The sub-tools of ActivAlignment together with other tools in a measuring application.

The status of the alignment (e.g., Success or Failure) and, after a successful alignment, the movement of the anchor relative to the reference image, can be sent to ActivDecision to evaluate them further and to *output tools* like ActivFile or ActivDigitalIO; furthermore, you can access results via the programming interface (see [section 4.3](#) on page 28).

1.2 The Sub-Tools of ActivAlignment

Besides its *master tool*, ActivAlignment provides 2 *support tools*. In [figure 1.3](#) they are depicted in an example ActivVisionTools application, where they are used to align a measuring ROI.

AVTAlignment is the *master tool* of ActivAlignment. Note that this ActiveX control does not



have a graphical user interface; thus, it is represented by its icon at design time and invisible at run time. If you forget to add it to the form and only add the support tools, they are disabled.



AVTAlignmentTeach is a *support tool* of ActivAlignment. It allows you to teach the patterns to which the ROIs of other ActivVisionTools are to be aligned to. How to use AVTAlignmentTeach is described in [section 2.2](#) on page [10](#).



AVTAlignmentUse is a *support tool* of ActivAlignment. It allows to specify the search space for the pattern and to select what is to be aligned and displayed. How to use AVTAlignmentUse is described in [section 2.3](#) on page [12](#) and [section 3.1](#) on page [16](#).

Chapter 2

Using ActivAlignment

This chapter will explain how to specify an alignment anchor and restrict the search space. The actual alignment of ROIs of other ActivVisionTools is the topic of the next chapter.

The corresponding Visual Basic projects show how to successfully recognize an IC in a sequence of images.




2.1	Selecting the Alignment Anchor	8
2.2	Teaching the Alignment Anchor	10
2.3	Finding the Anchor Again	12

2.1 Selecting the Alignment Anchor Using


You select an alignment anchor by creating regions of interest around it using AVTViewROI, which is a *support tool* of ActivView. ActivAlignment lets you choose between different shapes of ROIs, e.g., arbitrarily oriented rectangles or ellipses.

Visual Basic Example

Preparation for the following example:

- Open the project rois\alignment_rois.vbp. Alternatively, create a new project and place AVTView, AVTViewROI, and AVTAlignment on the form by double-clicking the icons , , and , respectively, with the left mouse button. Note that AVTAlignment is only represented by its icon on the form!
- Execute the application (Run ▸ Start or via the corresponding button). In this mode AVTAlignment is invisible.
- Open AVTViewFG by clicking into AVTView with the right mouse button and selecting Image Acquisition in the popup menu. Select the image board\board_01 in the combo box Input File.

The following steps are visualized in [figure 2.1](#).

- ① First, you have to tell AVTViewROI to create an ROI for ActivAlignment by selecting the corresponding entry in the combo box ActivVisionTool. In this box, all ActivVisionTools are listed that have been placed upon the form. By default, the tools are referenced by the name of the corresponding ActiveX control plus a counter to distinguish between multiple instances of a control on the form. In our example, there is only one item in the combo box, AVTAlignment1.
- ② Next, select the desired shape of the ROI, for example an arbitrarily oriented rectangle.
- ③ To draw the ROI, move the mouse in the image while keeping the left mouse button pressed. Please experiment at this point with the different shapes. The creation of a polygonal ROI () is more complex: By the first mouse movement, the first side of the polygon is created. You can add a new corner point by clicking on the line with the left mouse button; when you drag it by keeping the mouse button pressed, new polygon sides are created, where you can again add new corner points. To delete a corner point, drag it onto a neighboring corner point.
- ④ You can now move the ROI by dragging its pick point in the middle. By dragging the outer pick points you modify its shape. Again, please experiment to get familiar with the ROIs. Note that for a polygonal ROI the “middle point” appears at the center of gravity of the ROI and therefore changes whenever the polygon is modified; besides, it

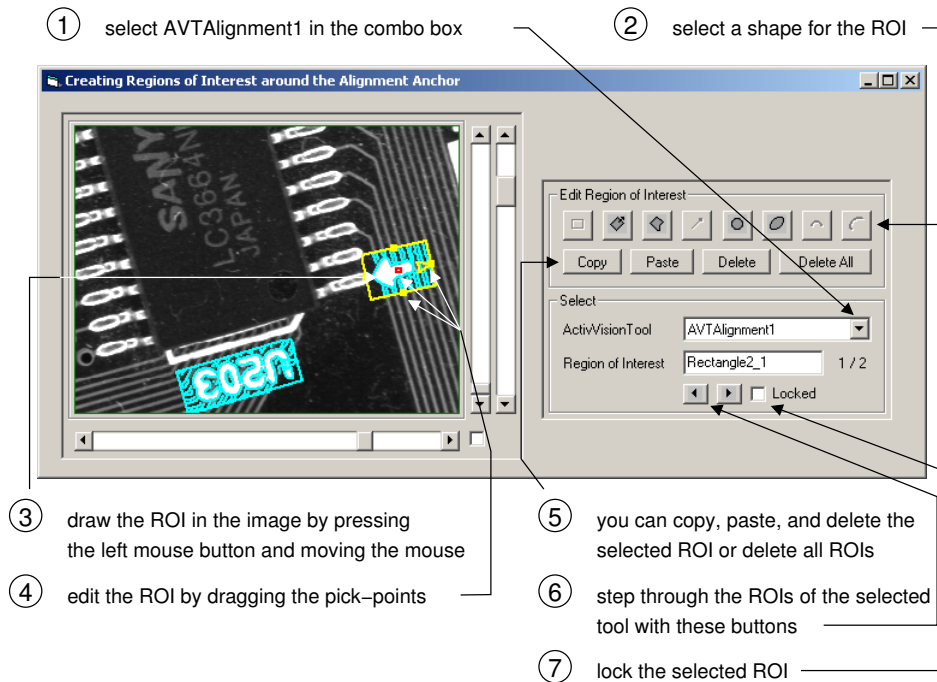




Figure 2.1: Creating regions of interest around the alignment anchor.

may even lie outside the ROI for concave polygons.

- ⑤ To create a second ROI, repeat step ②. You can also copy, paste, and delete the selected ROI or delete all ROIs.
- ⑥ ROIs can be selected by clicking next to it in the image. Alternatively, use the two arrow buttons   to step through all the ROIs of the selected tool.
- ⑦ By checking ☒ Locked you can lock the selected ROI and thus prevent it from accidental editing. Note that alignment ROIs are automatically locked after training the anchor (see the following section).



If you create multiple ROIs for ActiveAlignment, they are internally fused to form one anchor region consisting of separate sub-regions.

2.2 Teaching the Alignment Anchor Using

After selecting one or more regions that are to serve as the anchor for alignment, you actually teach the anchor using `AVTAlignmentTeach`.

Visual Basic Example

Preparation for the following example:

- If you worked on the previous example, you may continue using this project. At design time, add `AVTAlignmentTeach` and `AVTViewStatus` by double-clicking  and . You may delete `AVTViewROI`, as it can be opened at run time by clicking into `AVTView` with the right mouse button and selecting Region of Interest. Otherwise, open the project `training\alignment_training.vbp`.
- Execute the application (Run > Start or via the corresponding button). Open `AVTViewFG` by clicking into `AVTView` with the right mouse button and selecting Image Acquisition in the popup menu. Select the image `board\board_01` in the combo box Input File.

The following steps are visualized in [figure 2.2](#).

- ① You can specify a minimum contrast for contours with the slider `Contrast` or by directly typing the value in the text field to the right of the slider. Alternatively, click `Auto` to let `ActivAlignment` estimate the contrast value automatically. The chosen value is valid for all ROIs of `ActivAlignment`.
- ② If you know that the object will not appear in all possible orientations you may limit the corresponding range of rotation with the slider `Angle Range` or by directly typing the value in the text field to the right of the slider. As mentioned in [section 1.1](#) on page 2, you must restrict the range of rotation if the anchor shows any symmetries.

Note that you can also limit the range of rotation later in the application phase, but you cannot use a larger range than the one you specified in the training phase! The main disadvantage of teaching a large range of rotation is that the size of the internally stored training information grows proportionally with the range of rotation; as a direct result of this not only the training itself takes more time but also the saving or loading of training data. Furthermore, if the training data gets too large to fit into (virtual) memory, the application will be slowed down noticeably by the operating system. In such a case you must limit the range of rotation in the training phase.

- ③ In the frame `Display` you can choose whether the anchor region and the angle range are displayed in `AVTView`.
- ④ By clicking `Apply Training` you start the training process, which may take a while

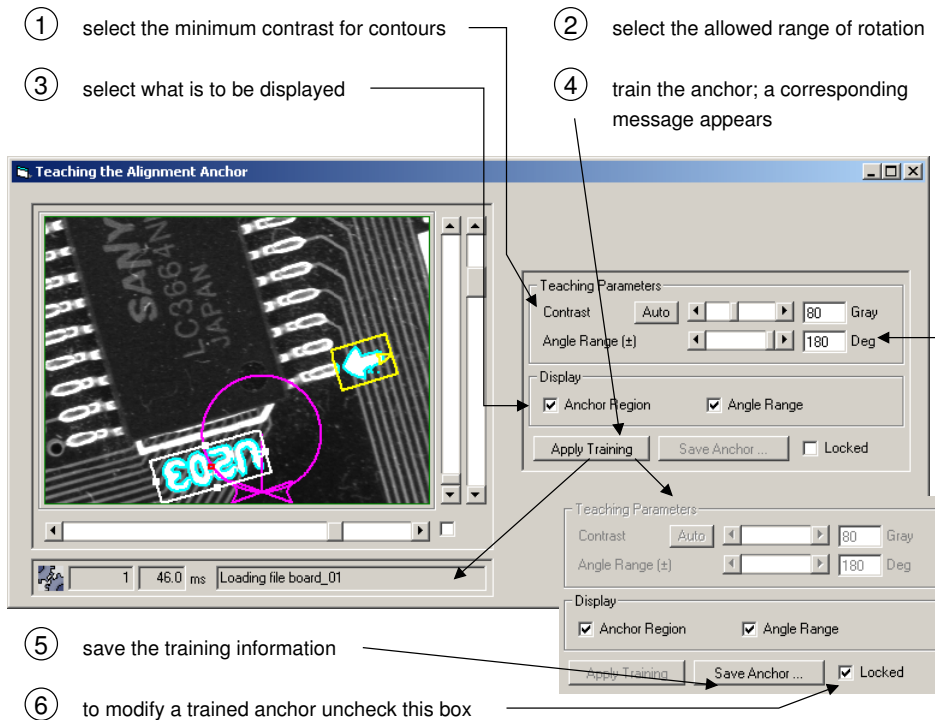


Figure 2.2: Teaching the alignment anchor.

to complete. During this time, AVTViewStatus displays a corresponding message. Afterwards, the appearance of AVTAlignmentTeach changes.

- ⑤ When you stop the application, the training information is automatically saved in a file with the extension `.alm`; the file name is composed of the names of the application and of the instance of AVTAlignment, e.g., `alignment_training_AVTAlignment1.alm` for this example project. Whenever you start the application again, the alignment file is automatically loaded. If you want to use the training information for other projects as well, you can save it in a separate file by clicking **Save Anchor** and specifying a file name in the appearing file selection box.
- ⑥ If you want to modify a trained anchor, e.g., modify the region of interest, **you must “unlock” it first** by unchecking the box ☒ Locked. Note that the anchor is unlocked automatically when you create another ROI for ActivAlignment.




2.3 Finding the Anchor Again Using

AVTAlignmentUse allows to configure the application phase of ActivAlignment by defining the search space and specifying what is to be displayed in AVTView. Furthermore, it lets you select whether ROIs are to be aligned and what happens in case the anchor is not found in an image; this is described in the following chapter.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. At design time, add AVTAlignmentUse by double-clicking . You may remove AVTAlignmentTeach as it can be opened by clicking into AVTView with the right mouse button and selecting Alignment Teach in the popup menu.
Otherwise, open the project matching\alignment_matching.vbp.
- ☐ Execute the application and load the image sequence board\board1.seq.

The following steps are visualized in [figure 2.3](#).

- ① As described in the previous section, when you start an application in which you already trained an anchor the training information is automatically loaded. Via a click on Load Anchor you can load a training file explicitly. Thus, you can reuse training information for multiple applications.
- ② When you click Set Reference, the current image is used as the reference image.
- ③ If you know that the object will not appear in all possible orientations you may limit the corresponding range of rotation with the slider Angle Range or by directly typing the value in the text field to the right of the slider. Note that you cannot use a larger range than the one you specified in the training phase (see [section 2.2](#) on page 10).
- ④ With the slider Search Region you can restrict the *search range*, i.e., the part of the image where the anchor is searched. The search range is defined by expanding the anchor region(s) by the amount of pixels selected with the slider.
- ⑤ With the slider Minimum Visible you can specify how much of the anchor must be visible at least. A part of the anchor may be “invisible” because it is occluded (or outside the search range) but also because of image disturbances, e.g., noise.
- ⑥ The slider Speedup allows to trade safety for speed. If you set it to 0, ActivAlignment applies a very thorough search algorithm that guarantees that the anchor is found if it is within the search space. The higher the speedup factor the faster but also rougher the search will be, with the possible effect that ActivAlignment fails to find an anchor even

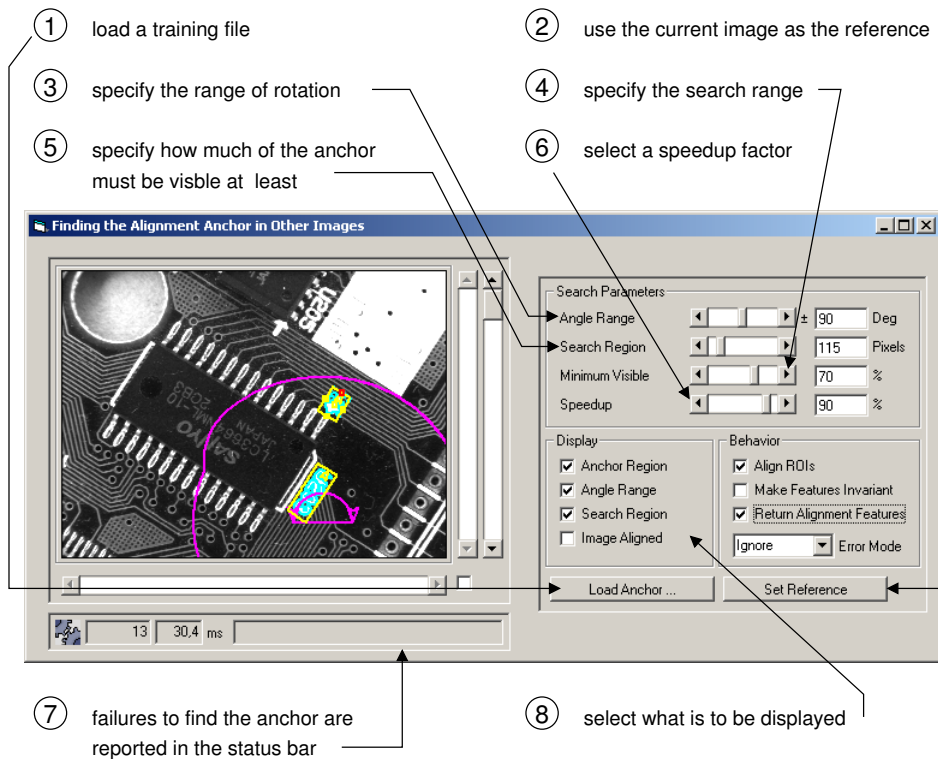


Figure 2.3: Finding the trained anchor in other images.

if it is within the search space.

- ⑦ To get to know the effect of the parameters described above, step through the image sequence by clicking **Single** in AVTViewFG. If ActivAlignment fails to find the anchor a corresponding error message appears in AVTViewStatus.
- ⑧ In the frame Display you can select what is to be displayed in AVTView. The use of most check boxes is obvious; if you check ☒ Image Aligned, the whole image including ROIs and results is transformed such that the anchor is positioned as it was in the reference image.

The parameters in the frame Behavior are described in [section 3.1](#) on page 16.

Chapter 3

Combining ActiveAlignment with other ActiveVisionTools

While the previous chapter explained how to create an alignment anchor and to configure the search space, this chapter shows how to actually align regions of interest of other Activision-Tools and how to evaluate and output their results. How to access results via the programming interface is described in [chapter 4](#) on page 23.

In the corresponding Visual Basic projects, **ActivMeasure** is used to inspect the pins of an IC. Please consult the [User's Manual for ActivMeasure](#) for detailed information about this tool.

3.1	Aligning the ROIs for ActivMeasure	16
3.2	Evaluating Results	18
3.3	Output of Results	20

3.1 Aligning the ROIs for ActivMeasure

If alignment information is present in an ActivVisionTools application, it is used automatically to align the regions of interest of other ActivVisionTools. AVTAlignmentUse allows you to further configure the alignment behavior, e.g., whether results are to be re-aligned to the reference image or what is to happen if ActivAlignment fails to find the anchor.

In the example, ActivMeasure is used to inspect the pins of an IC regarding their width and distance.

Visual Basic Example

Preparation for the following example:

- ☐ We recommend to open the project `aligning\alignment_aligning.vbp`, as it already contains a suitable anchor; furthermore, the measuring parameters are already set up.
- ☐ Execute the application (Run ▷ Start or via the corresponding button) and load the image sequence `ic\ic1.seq`; AVTViewFG can be opened by clicking into AVTView with the right mouse button and selecting Image Acquisition.

The following steps are visualized in [figure 3.1](#).

- ① To create an ROI for ActivMeasure, open AVTViewROI by clicking into AVTView with the right mouse button and selecting Region of Interest. Select AVTMeasure1 in the combo box ActivVisionTool and create a line-shaped ROI over the pins of the IC as shown in [figure 3.1](#). Automatically, the measured width of the pins, the distance between them, and the position of their centers is displayed in ActivDataView.
- ② Check ☒ Align ROIs to let the ROI be aligned. If you step through the image sequence by clicking `Single` in AVTViewFG, the measuring ROI should now always lie over the pins.
- ③ If you check ☒ Make Features Invariant, the results of ActivMeasure are transformed back into the reference image (see [section 1.1](#) on page 2). The effect can be seen clearly in the column Row Center: In the reference image, the pins are lined up horizontally, therefore their respective Row Center varies only by one or two pixels. If you switch to an image where the chip is rotated, the values for Row Center will vary strongly before checking the box, and correspond to the ones of the reference image afterwards.
- ④ For one image of the sequence ActivAlignment fails to find the anchor because of the limited angle range. In the combo box Error Mode, you can choose the desired reaction: If you select 'Ignore', ROIs are not aligned, i.e., the reference position is used.

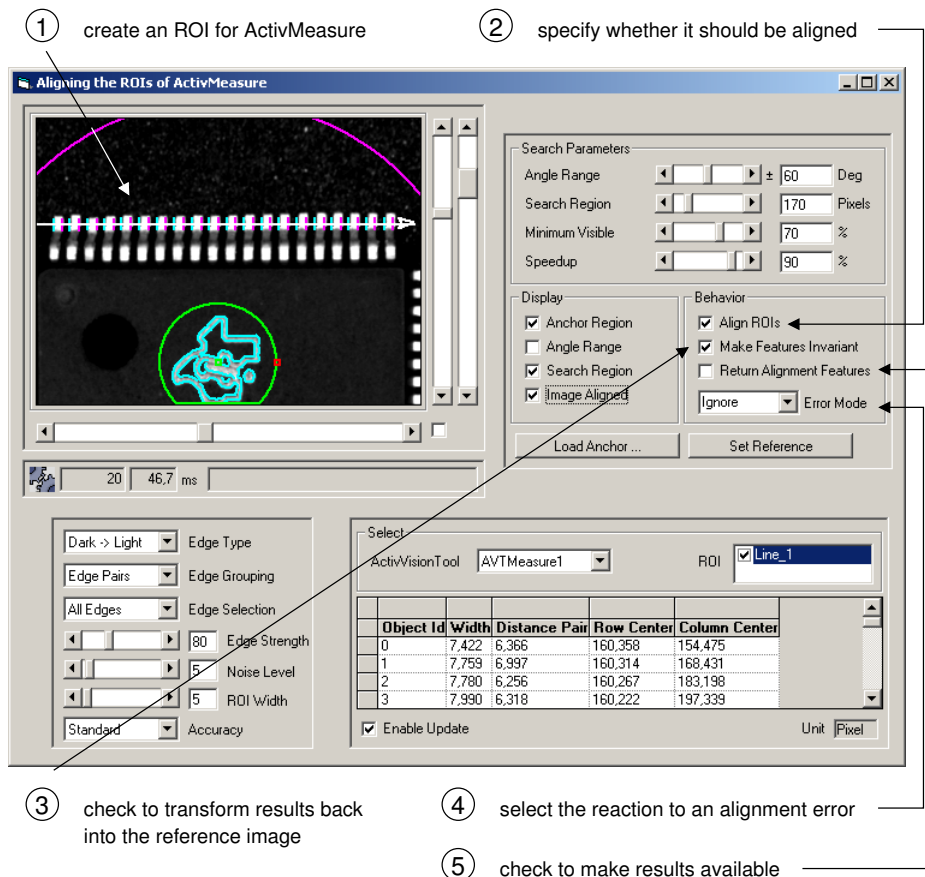


Figure 3.1: Aligning a measuring ROI to an IC.

In contrast, in the mode 'Fail Feature' ActivMeasure does not perform any image processing. If you select 'Stop Execution', the application stops before executing any other tools.

- ⑤ If you check ☒ Return Alignment Features, the results of ActivAlignment, i.e., its status (e.g., Success or Failure) and, after a successful alignment, the movement of the anchor relative to the reference image, is made available to tools like ActivDecision or ActivFile. Note that the alignment results are automatically sent to other ActivVision-Tools whose ROIs are aligned, i.e., without requiring any action on your part.


3.2 Evaluating Results Using

In the previous example, you have employed ActivVisionTools to extract and display measurement results. Using ActivDecision, you can now evaluate these results by formulating *conditions* the results have to meet in order to be “okay”. For a detailed description of ActivDecision please consult the [User’s Manual for ActivDecision](#).

We continue with the example task of inspecting the pins of an IC. To be accepted as “okay”, the pins must be 7.0–8.5 pixels wide, their distance should be 5.2–8.7 pixels.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. At design time, add AVTDecision to the form by double-clicking  with the left mouse button.
Otherwise, open the project decisions\alignment_decisions.vbp.
- ☐ Execute the application and load the image sequence ic\ic1.seq.

The following steps are visualized in [figure 3.2](#) (only AVTDecision shown).

- ① The main functionality of ActivDecision is presented by its *support tools*, which can be opened via clicking on AVTDecision with the right mouse button. The *master tool* displays the overall evaluation of the application.
- ② To view the current feature and evaluation results check ☒ Enable Update in AVT-DecisionViewResults.
- ③ ActivDecision lets you compare the value of an individual object, ROI, or tool feature with two boundary values, a minimum and a maximum value. You can formulate conditions for features by specifying values in the columns Min and Max and selecting a comparison mode in the column Operation. If you select None, the feature is not evaluated. [Figure 3.2](#) shows suitable conditions to check that all 20 pins are present and “okay”.
- ④ Those features that meet their condition appear in green, the others in red. If at least one feature is “not okay”, the whole object, ROI, or tool is evaluated as “not okay” as well. Analogously, the overall evaluation of application, which is visualized by AVT-Decision, depends on the tool evaluations. Step through the image sequence by clicking in AVTViewFG and examine the evaluations. In some of the images, pins are bent, resulting in a wrong width or distance.
- ⑤ In the example application, all pins should meet the same condition. Instead of spec-

1 open the support tools via the context menu (right mouse click)

2 first, enable the update of results

3 formulate conditions for objects, ROIs or tools

4 the evaluations are displayed immediately

5 specify default conditions

6 check this box to show the used parameters

Failed

Evaluating Results Using ActivDecision

Path: AVTMeasure1

Tool ROI Object	Name	Value	Min	Max	Interpretation	Operation
AVTMeasure1	Objects Tool	19	0	100	Number	None
	Good Objects Tool	18	0	100	Number	None
	Bad Objects Tool	1	0	100	Number	None
	Good ROIs	0	0	10	Number	None
	Bad ROIs	1	0	0	Number	= Max
	Objects ROI	19	20	20	Number	= Max
	Good Objects ROI	18	0	100	Number	None
	Bad Objects ROI	1	0	0	Number	= Max
	Width	14,751	7,000	8,500	Number	Inside
	Distance Pair	6,390	5,200	8,700	Number	Inside
	Width	7,669	7,000	8,500	Number	Inside
	Distance Pair	6,357	5,200	8,700	Number	Inside

Enable Update ☒ Fit Cell Size ☒ Substitute Default ☒ Unit: Pixel

AVTDecisionViewDefaults

Path: AVTMeasure1

Tool ROI Object	Name	Min	Max	Interpretation	Operation
AVTMeasure1	Objects ROI	20	20	Number	= Max
	Good Objects ROI	0	100	Number	None
	Bad Objects ROI	0	0	Number	= Max
	Width	7,000	8,500	Number	Inside
	Distance Pair	5,200	8,700	Number	Inside
Line_1	Width	Default	Default	Default	Default
	Distance Pair	Default	Default	Default	Default

Fit Cell Size ☒ Substitute Default ☐ Unit: Pixel

Figure 3.2: Formulating conditions to evaluate the measurements.

ifying the same conditions for each object, you can specify default conditions using AVTDecisionViewDefaults. Defaults can be set per tool or per ROI; ROI defaults override tool defaults, and individual conditions override defaults.

- ⑥ If you check ☒ Substitute Default, the entries marked Default are substituted by their actual content.


To check the success of the alignment itself you can formulate a condition for ActivAlignment's feature Status, e.g., that it must be equal to Success. If you set the Angle Range to 60, the alignment fails for one image of the sequence.

3.3 Output of Results Using

Using ActivFile, you can write the results and the evaluations to a log file. How to access results via the programming interface is described in [section 4.3](#) on page 28, how to output them via a serial interface or a digital I/O board in the [User's Manual for ActivSerial](#) and the [User's Manual for ActivDigitalIO](#), respectively.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. At design time, add AVTOutputFile by double-clicking . Otherwise, open the project output\alignment_output.vbp.
- ☐ Execute the application and load the image sequence ic\ic1.seq.

The following steps are visualized in [figure 3.3](#) (only sub-tools of ActivFile shown).

- ① By clicking on **Select**, you can open a file selector box to choose a file name for the log file, which will then appear in the text field beside the button. By pressing **Clear File**, you can clear the content of the selected file.
- ② By checking ☒ **Enable Writing** you enable the writing mode.
- ③ You can open the ActivFile's two dialogs **DialogFileOptions** and **DialogOutputDataSelect** by clicking **File Options** and **Data Selection**, respectively.
- ④ In **DialogFileOptions**, you can choose between two file formats: Standard text (suffix .txt) and the so-called *comma-separated values* files (suffix .csv), which can be used as an input to Microsoft Excel. Furthermore, you can select a delimiter.
- ⑤ In the same dialog you can limit the size of the log file in form of the number of *cycles* that are to be recorded. A cycle corresponds to one processing cycle from image input to the evaluation and output of results. If you use this option, ActivFile creates two log files and switches between them, thus assuring that you can always access (at least) the results of the last N cycles, N being the specified number of cycles.
- ⑥ By pressing **Estimate**, you can let ActivFile estimate the size of one cycle. Note that you must first select the output data in order to get meaningful results!
- ⑦ In the left part of **DialogOutputDataSelect**, you can navigate through the result hierarchy similarly to ActivDecision.
- ⑧ In the other parts of **DialogOutputDataSelect**, choose the output data by checking the corresponding boxes. You may output different items depending on the evaluation of

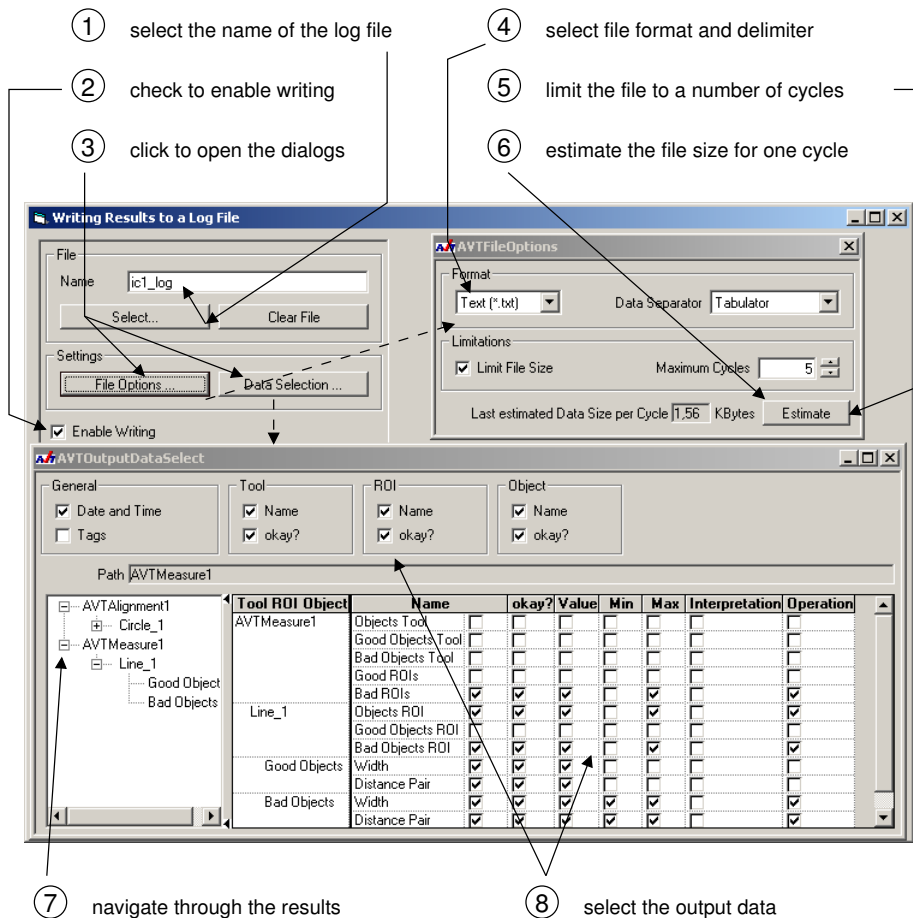


Figure 3.3: Customizing log files.

an object. By clicking on the column labels with the right mouse button you can check or uncheck all boxes in the column; similarly, you can check or uncheck whole rows or all rows of a certain tool.

If you now step through the image sequence by clicking **Single** in AVTViewFG, the log file is created. [Figure 3.4](#) shows part of an example log file, including a failed alignment.

```

09/23/04 14:38:56
AVTMeasure1 no
    Bad ROIs      no  1  0  = Max
    Line_1 no
        Objects ROI      no  0  20  = Max
        Bad Objects ROI  yes  0  0  = Max
AVTAlignment1 yes
    Status Failure

09/23/04 14:38:57
AVTMeasure1 no
    Bad ROIs      no  1  0  = Max
    Line_1 no
        Objects ROI      yes 20 20  = Max
        Bad Objects ROI  no  1  0  = Max
    0 yes
        Width yes 7.268505
        Distance Pair yes 7.230632
    1 yes
        Width yes 7.719226
        Distance Pair yes 7.030830
    2 yes
        Width yes 7.818486
        Distance Pair yes 6.217172
    3 yes
        Width yes 7.998314
        Distance Pair yes 6.348477

(...)

    17 yes
        Width yes 7.879461
        Distance Pair yes 6.929342
    18 no
        Width yes 7.604921 7.000000 8.500000 Inside
        Distance Pair no 12.204086 5.200000 8.700000 Inside
    19 yes
        Width yes 7.614961
AVTAlignment1 yes
    Status Success

```

Figure 3.4: Part of an example log file.

Chapter 4

Tips & Tricks

This chapter contains additional information that facilitates working with ActivAlignment, e.g., how to modify the graphical display of results and how to customize the appearance of an ActivAlignment application in the two execution modes. Furthermore, it shows how to access results directly via the programming interface of ActivVisionTools.

4.1

Adapting the Display of Results

24

4.2

Configuring the Two Execution Modes

26

4.3

Accessing Results Via the Programming Interface

28

4.3.1

Basic Principles

29

4.3.2

An Example Project

30

4.1 Adapting the Display of Results Using

You can adapt the way results and ROIs are displayed using `AVTViewDisplayModes`, which is a *support tool* of `ActivView`.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the example in the previous chapter, you may continue using this project.
Otherwise, open the project `display\alignment_display.vbp` and execute it (Run > Start or via the corresponding button).
- ☐ Load the image sequence `ic\ic1.seq` and create additional ROIs for `ActivMeasure`; `AVTViewFG` and `AVTViewROI` can be opened at run time via a right mouse button click on `AVTView`.

The following steps are visualized in [figure 4.1](#) (not shown: `AVTMeasure`). Experiment by choosing different settings for the display parameters and watching the result.

- ① Open `AVTViewDisplayModes` by clicking on `AVTView` with the right mouse button and selecting `Display Modes` in the popup menu.
- ② Change the color of the selected pick point. Select another pick point by clicking into its vicinity.
- ③ Change the color of the currently selected ROI. Select another ROI by clicking into its vicinity. Change the color of the other, not selected ROIs of the currently selected tool. Change the color of the ROIs of the other, not selected tools. Select another tool in the combo box `ActivVisionTool` of `AVTViewROI` (if this box contains more than one item). If you use `ActivDecision` to evaluate results, it can mark ROIs evaluated as “not okay” in a special color.
- ④ Change the LUT (look-up table) that is used to display the image.
- ⑤ Change the line width of the ROIs or of the results (edges for `ActivMeasure`, contours of the anchor and search and angle range for `ActivAlignment`).
- ⑥ The `ActivVisionTools` assign results to two groups to allow to display them using different colors. The contours of the alignment anchor (`ActivAlignment`) and the first edge of a pair (`ActivMeasure`) belong to group I. Change the color of this group.
- ⑦ The search and the angle range (`ActivAlignment`) and the second edge of a pair (`ActivMeasure`) belong to group II. Change the color of this group.

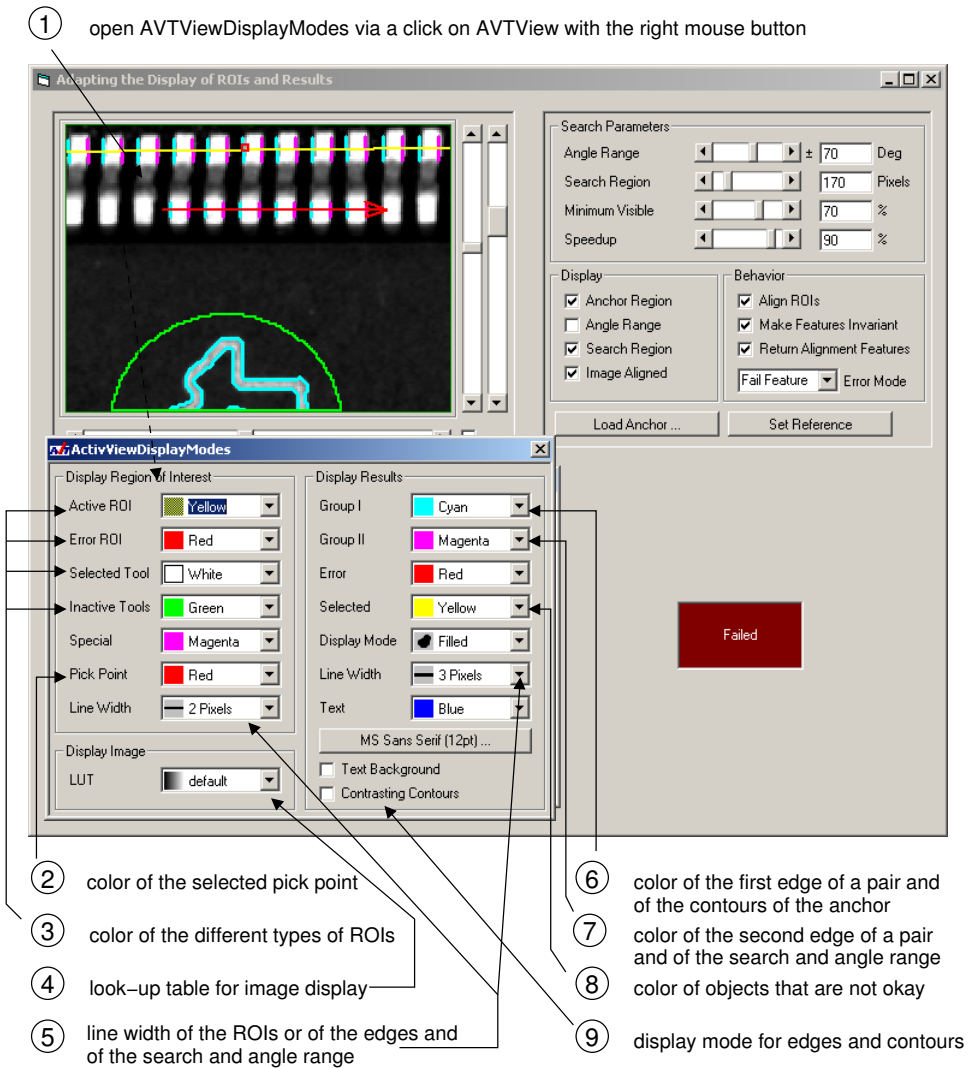


Figure 4.1: Adapting the display of ROIs and results.



- ⑧ Change the color ActivDecision uses to mark objects evaluated as “not okay”.
- ⑨ If you check this box, the edges and contours are framed with a contrasting color to make them more visible against busy backgrounds.

4.2 Configuring the Two Execution Modes Via and


In an ActivVisionTools application you can switch between two execution modes: the *configuration mode* and the *application mode*. The former should be used to setup and configure an application, the latter to run it. ActivView's *support tools* AVTViewExecute and AVTViewConfigExec allow you to switch between the two modes and to customize the behavior of an ActivVisionTools application in the two execution modes, e.g., display live images only in the *configuration mode* to setup your application, but then switch it off in the *application mode* to speed up the application. A third sub-tool, AVTViewExecuteSimple, provides a single button to execute the application. In this section we also describe AVTViewStatus in more detail.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. At design time, add AVTViewExecuteSimple and AVTViewStatus to the form by double-clicking the icons  and  with the left mouse button.
Otherwise, open the project usermodes\alignment_usermodes.vbp.
- ☐ Execute the application and load the image sequence ic\ic1.seq.

The following steps are visualized in [figure 4.2](#).

- ① Open AVTViewExecute and AVTViewConfigExec by clicking on AVTView with the right mouse button and selecting Execution and Execution Parameters.
- ② Switch between the two execution modes via AVTViewExecute's combo box Mode.
- ③ To execute one cycle, press **Single**. With the other two buttons you can let the application run continuously and stop it again. By default, AVTViewExecuteSimple starts and stops an application; how to change its behavior to a single-step button is described in the User's Manual for ActivView, [section 3.4](#) on page 34.
- ④ For each of the two execution modes, you can choose what is to be displayed by checking the corresponding boxes in AVTViewConfigExec. Furthermore, you can specify if images can be dragged to the image window and whether ROIs can be modified in the two modes; by default, this is disabled in the *application mode* to prevent you from accidentally moving or deleting an ROI.
- ⑤ In AVTViewStatus, an icon indicates the current execution mode of the application. In the mode , the application does not perform any processing and waits for your interaction. If you start the continuous mode the cogwheels rotate; any interaction on

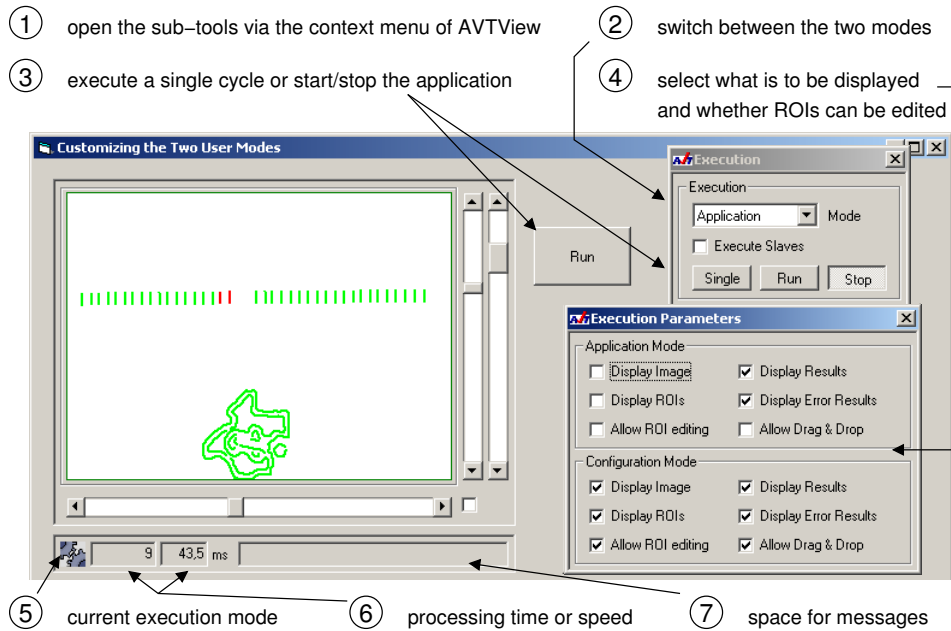


Figure 4.2: Customizing and switching between the two execution modes.

your part is stored in the event queue and processed after the current cycle is finished. If the cursor gets “busy”, the ActivVisionTools have started a particularly time-consuming operation, e.g., connecting to an image acquisition device. Any interaction on your part is then deferred to the end of this operation.

- ⑥ AVTViewStatus also shows the number of processed cycles and the time needed for the last processing cycle.
- ⑦ AVTViewStatus display two types of messages: Informative messages describe, e.g., what the application is doing while it is busy, while error messages indicate errors that prevent the application from working correctly, e.g., the failure to detect the bar code type automatically. Note that the failure to find an anchor ranks as an error only if you selected 'Stop Execution' in the combo box Error Mode.

If AVTViewStatus is not added to an application, error messages are displayed in popup dialogs.

More information about AVTViewStatus, e.g., how to modify its appearance, can be found in the User's Manual for ActivView, [section 3.3](#) on page 32.

4.3 Accessing Results Via the Programming Interface

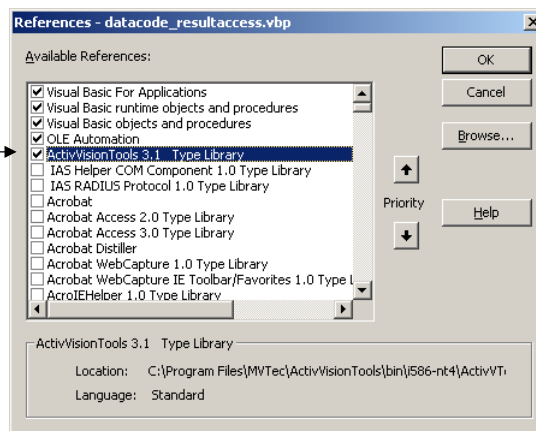
The previous chapters and sections showed how to use ActivVisionTools interactively, i.e., via the graphical user interfaces presented by the underlying ActiveX controls. In this mode, you can develop the image processing part of your machine vision application rapidly and easily, without any programming. However, there is more to ActivVisionTools than the graphical user interfaces: Because ActivVisionTools comes as a set of ActiveX controls, it provides you with an open programming interface, thereby offering full flexibility.

In this section, we show how to access the results of alignment via the programming interface. With this, you can, e.g., realize an application-specific graphical user interface, perform additional processing on the results, or send results to a special output device. More examples showing how to use the programming interface, e.g., to access evaluation results, can be found in other manuals, e.g., in the User's Manual for ActivMeasure in [section 4.3](#) on page 30 or in the User's Manual for ActivDecision in [section 3.4.1](#) on page 31. Detailed information about the programming interface can be found in the **Reference Manual**.

As in the previous sections, the examples stem from Visual Basic 6.0; if the (ActivVisionTools-specific) code differs in Visual Basic .NET, the corresponding lines are also shown (for the first appearance only). For other .NET languages or C++, please refer to the Advanced User's Guide for ActivVisionTools, [section 1.2.3](#) on page 5 and [section 1.3.4](#) on page 28, respectively. Please note that we assume that readers of this part have at least a basic knowledge of Visual Basic.



To work with the programming interface, in VB 6.0 you must first **add the ActivVisionTools type library to the project's references** by checking the box labeled ActivVisionTools Type Library in the menu dialog Project > References. In Visual Basic .NET, the reference is added automatically.



4.3.1 Basic Principles

The principal idea behind accessing the results of an `ActivVisionTool` is quite simple: When a tool has finished its execution, it raises an event called `Finish`, sending its results as a parameter. If you want to access the results, all you have to do, therefore, is to create a corresponding event procedure, which handles the event.

Within the Visual Basic environment, you can create event procedures very easily as shown in [figure 4.3](#): In the header of the code window corresponding to a form there are two combo boxes. Select the instance of `AVTAlignment` (by default called `AVTAlignment1`) in the left combo box. The right combo box then lists all events available for this object; when you select `Finish`, the event procedure is created automatically.

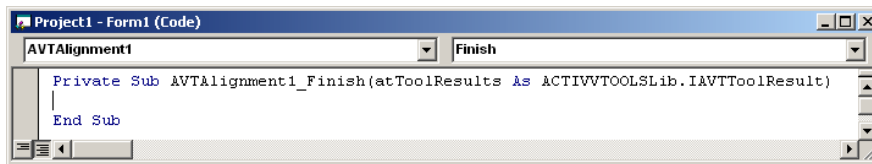


Figure 4.3: Creating a procedure to handle the event `Finish`.

In Visual Basic .NET, the event handler has a different signature:

```
Private Sub AxAVTAlignment1_Finish(ByVal sender As Object, _
                                   ByVal e As _
                                   AxActivVTools.__AVTAlignment_FinishEvent) _
    Handles AxAVTAlignment1.Finish
```

A first difference is that tool names start the prefix `Ax`, i.e., `AVTAlignment` becomes `AxAVTAlignment`. The main difference, however, is that the tool results are not directly passed; instead, they are encapsulated in the parameter `e`. From there, they can be extracted with the following lines:

```
Dim atToolResults As AVTToolResult
atToolResults = e.atToolResults
```

To use classes like `ACTIVVTOOLSlib.AVTToolResult` without the namespace `ACTIVVTOOLSlib` as in the code above, you must import this namespace by inserting the following line at the very beginning of the code (more information about importing namespaces can be found in the Advanced User's Guide for `ActivVisionTools` in [section 1.2.4.5](#) on page 12):

```
Imports ACTIVVTOOLSlib
```

`atToolResults` contains the alignment results, i.e., its `Status` (e.g., `Success` or `Failure`) and, after a successful alignment, the movement of the anchor relative to the reference image.

These results, called *features*, can be accessed by specifying their ID in a call to the method `atToolResults.Value(feature handle)`. The feature handles are available as functions of the corresponding tool, e.g., `AVTAlignment1.FeatureHandleStatus` being the handle for the status of the alignment.

The values of the feature `Status` are adapted via *Native Language Support* (see the manual [Getting Started with ActivVisionTools](#)) to the different languages. The used values are available via the programming interface. The following code shows how to access the value signaling a successful alignment and compare the current alignment status to it:

```
Dim handleStatus As Integer
Dim sStatusSuccess As String

sStatusSuccess = AVTAlignment1.NLS.Text("StatusSuccess")

handleStatus = AVTAlignment1.FeatureHandleStatus
If atToolResults.Value(handleStatus) = sStatusSuccess Then
    ' some reaction
End If
```

Note that in contrast to tools like `ActivMeasure`, the results of `ActivAlignment` are available at tool level; the ROIs themselves have no results.

4.3.2 An Example Project

The `ActivVisionTools` distribution includes the example Visual Basic project `resultaccess\alignment_resultaccess.vbp`, which uses the methods described in the previous section to extend the application of [chapter 3](#) on page 15. The example project has been configured in such a way, that the alignment fails for one image of the image sequence `ic\ic1.seq`. This is checked and signaled by an error message. By increasing the parameter `Angle Range` in `AVTAlignmentUse`, you can “repair” the alignment, which is also signaled by the application.

Besides accessing the alignment status, the project code contains additional functionality, which is explained briefly in the following (only for Visual Basic 6.0!). If the alignment fails, the function `SetAlarm` stops the application and switches the color of the element beside the message list to red. The function `ClearAlarm` resets the color to green.

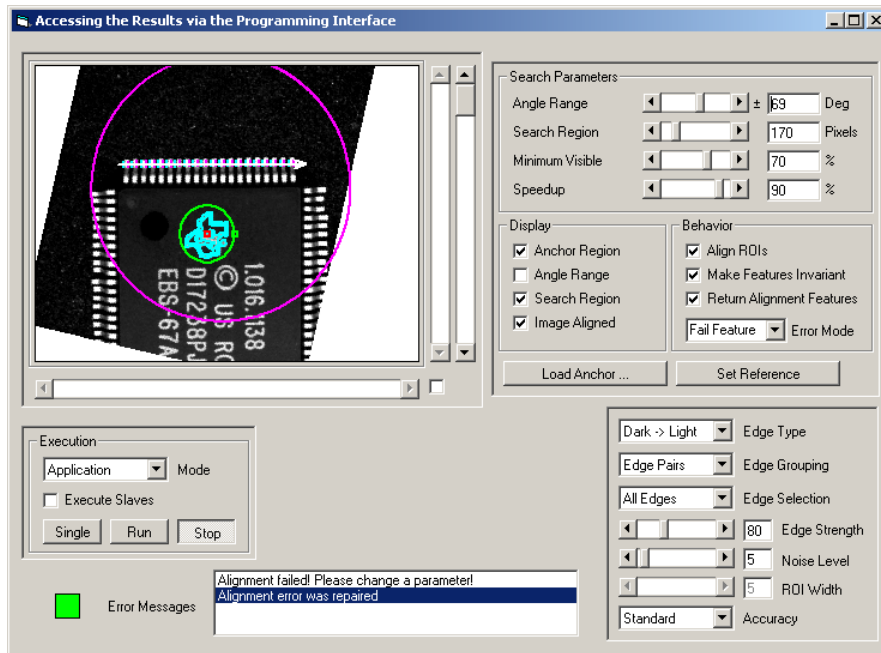


Figure 4.4: Accessing the alignment status.

```

Private bIsAligned As Boolean

If atToolResults.Value(handleStatus) = sStatusSuccess Then
    bIsAligned = True
    If bIsError = True Then
        Call ClearAlarm
    End If
Else
    sErrMsg = "Alignment failed! Please change a parameter!"
    Call DisplayMessage(sErrMsg)
    Call SetAlarm

    bIsAligned = False
End If

```

```
Private bIsError As Boolean

Private Function SetAlarm()
    AVTView1.RunState = False
    Light.BackColor = vbRed
    bIsError = True
End Function

Private Function ClearAlarm()
    Light.BackColor = vbGreen
    bIsError = False
End Function
```

One has to keep in mind that AVTAlignment is executed not only when the next image is grabbed but also whenever you modify its parameters. To distinguish the two cases an event raised by AVTView at the start of each execution cycle can be used to set a variable called bIsNewCycle:

```
Private bIsNewCycle As Boolean

Private Sub AVTView1_CycleStart()
    bIsNewCycle = True
End Sub
```

This variable is now used together with the variable bIsAligned, which stores the status of the previous alignment, to distinguish the following cases: If the alignment fails for a new image (bIsNewCycle = True, an error message is created. If the alignment fails while you are trying to find suitable parameters (bIsNewCycle = False and bIsAligned = False), nothing happens; a message appears only, when you have found a suitable parameter (bIsNewCycle = False and bIsAligned = False). Finally, if you modify a parameter such that a previously successful alignment now fails (bIsNewCycle = False and bIsAligned = True), this is also signaled by a message.

```

If atToolResults.Value(handleStatus) = sStatusSuccess Then
  If bIsAligned = False And bIsNewCycle = False Then
    sErrMsg = "Alignment error was repaired"
    Call DisplayMessage(sErrMsg)
  Else
    bIsNewCycle = False
  End If
  bIsAligned = True
  If bIsError = True Then
    Call ClearAlarm
  End If
Else
  If bIsNewCycle = True Then
    bIsNewCycle = False
    sErrMsg = "Alignment failed! Please change a parameter!"
    Call DisplayMessage(sErrMsg)
    Call SetAlarm
  Else
    If bIsAligned = True Then
      sErrMsg = "Alignment error because of changed parameter!"
      Call DisplayMessage(sErrMsg)
      Call SetAlarm
    End If
  End If
  bIsAligned = False
End If

```

When using the programming interface of ActivVisionTools, you leave the safe world of the graphical user interfaces where all input is checked for validity automatically. In contrast, if you try to access a non-existent result via the programming interface, a run-time error is caused, which halts your application. To avoid this, you can use the Visual Basic error handling mechanisms, i.e., set up an error handler that examines any occurring error and reacts in a suitable way. In the example project, if an error is caused by the result access a popup with the error description appears and the function `SetAlarm` is called. To view the effect of the error handler, switch of the creation of alignment features via the check box ☐ Return Alignment Features in `AVTAlignmentUse`.

```
Private Sub AVTAlignment1_Finish(atToolResults As _
                                ACTIVVTOOLSlib.IAVTToolResult)

    ' variable declarations

On Error GoTo ErrorHandler

    ' procedure body
Exit Sub

ErrorHandler:
    Dim sTitle As String
    If Left(Err.Source, 11) = "ActivVTools" Then
        sTitle = "ActivVisionTools Error"
    Else
        sTitle = "Runtime Error " & CStr(Err.Number)
    End If
    Call MsgBox(Err.Description, vbExclamation, sTitle)
    Call SetAlarm
End Sub
```

By placing the following code at the beginning of AVTAlignment1_Finish, the actual result access is restricted to the *application mode*. With this mechanism you can setup the vision part of your application in the configuration mode without having to worry about run-time errors.

```
If Not AVTView1.ExecutionMode = eApplicationMode Then
    Exit Sub
End If
```